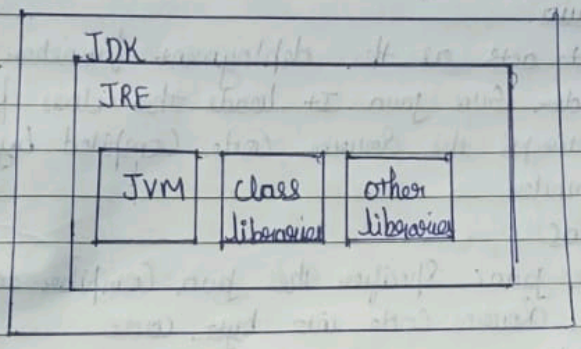
Assignment - 1

Q1 Write a short note on Java Development Kit (JDK)

Ans Java Development Kit is a bundle of Software development tools and Supporting libraries combined with the Java Runtime Environment (JRE) and Java Virtual Machine (JVM)



Java Virtual Machine
JVM is a Software tool responsible for creating a run-time environment for the Java Source code to run the very powerful feature of Java "Write Once and run anywhere" is made possible by JVM the JVM stays right on the top of the host operating System and converts the Java Source code into Bytecode (machine language) and execute

ing algorithm.

remotely.

policy.

d.

ticular node.

processing capability

policy.

Most of the load-balancing algorithms use single thre

Overloaded and underloaded. In this policy, if load

underloaded and it accepts new local or remote proces

nodes transfer the local processes and reject to accept

Overloaded

Thresh

the program

Java Run-Time Environment

JRE is a software platform where all the Java source code are executed JRE is responsible for integrating the software plugins, jar files and support libraries necessary for the source code to run

Components of JDK in Java

Java

It acts as the deployment launcher in the older SUN Java It loads the class files and interprets the source code compiled by the Java compiler

Javac

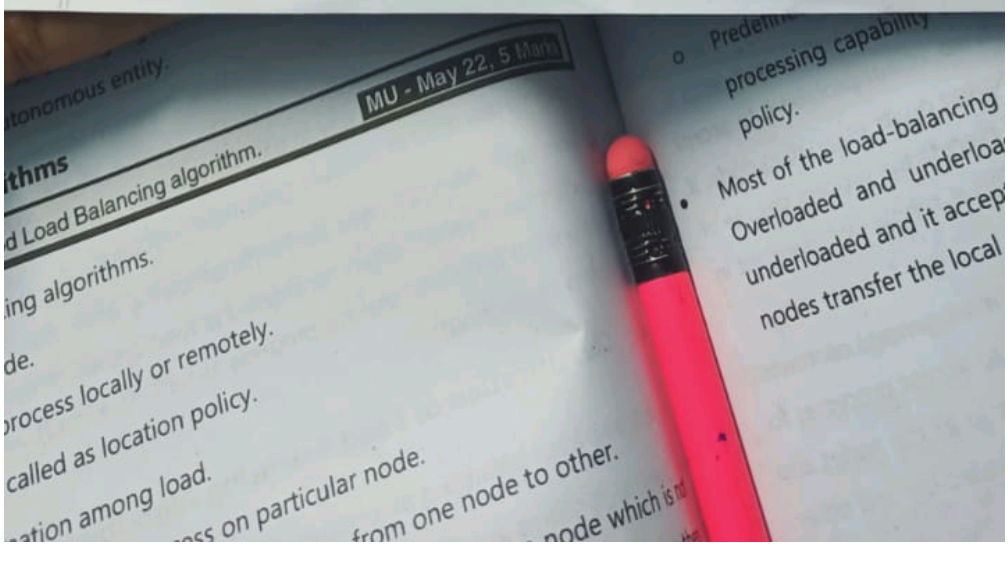
The javac specifies the Java compiler to convert the source code into byte code

Javadoc

The javadoc generates documentation for the comments added in the source code

Jar

The Jar helps the archives to manage the Jar files in the package library.



Q2. List and Explain the Salient features of the Java

Ans ① Simple
Java is designed to be easy to learn. Its Syntax is quite simple, clean and easy to understand. Java is inspired by C and C++ there is no need to remove unreferenced objects because there is an Automatic Garbage Collection in Java

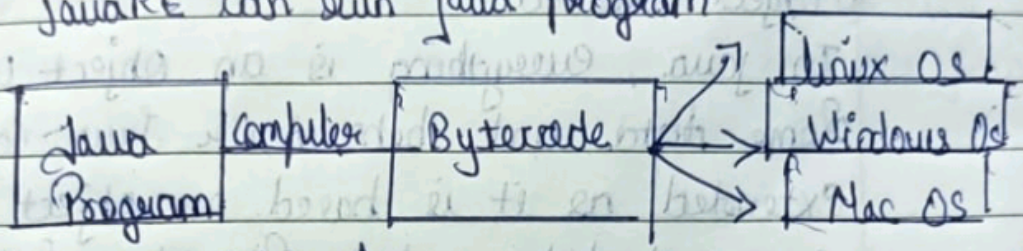
② Object Oriented
In java, everything is an object which has some data and behaviour. Java can be easily extended as it is based on object Model. OOPs is a methodology that simplifies Softwares development and maintainance by providing some rules. Everything in Java is written in terms of classes and objects. Basic Concept of OOPs

i) Object
ii) Class
iii) Inheritance
iv) Polymorphism
v) Abstraction
vi) Encapsulation

PAGE No.
DATE / /

③ Platform Independent

Java is a write Once, run anywhere language A platform is the hardware or software environment in which a program runs - Java provides a software based platform Java Code can be executed on multiple platforms. like windows, linux, Sun Solaris, mac os etc on Compilation Java program is compiled into bytecode this bytecode is platform independent and can be run on any machine, plus the bytecode format also provide Security Any Machine with JavaRE can run java program



④ Architecture - Neutral

Java Compiler generates an architecture - neutral object file format, which makes the compiled code executable On many processor, with the presence of Java runtime System

⑤ Portable

Being architecture - neutral and having no implementation dependent aspect of the Specification make Java portable Everything related to storage is predefined

⑥ Robust

Java uses Strong memory management

Q3 list and explain the Components of Java Virtual Machine

Ans Components of Java virtual Machine

① class loader

class loader is a Subsystem of JVM which is used to load class file whenever a "java" Source file is compiled, it is converted into byte code as a "class" file when this class is used, the class loader loads it into the main memory the first class to be loaded into memory is usually the class that contains the main() method there are three phases

* loader

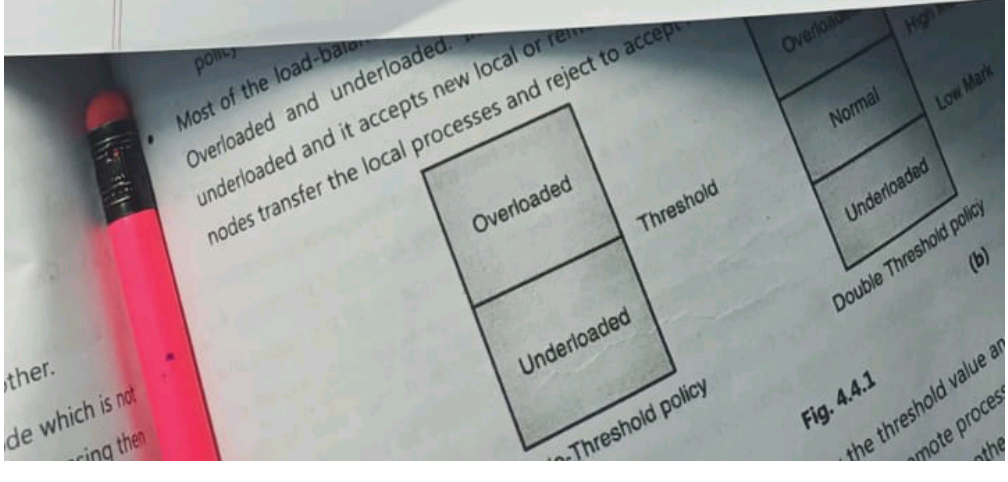
This phase loads files from Secondary memory into the main memory for execution loading involves taking the binary representation of a class or interface with a particular name and generating the original class or interface from that

* linking

linking is a class or interface involves combining the different elements and the dependencies of the program together linking included

* verification

This phase checks the Structural Correctness of the "class" file by checking it against a set of



Constraints or rules

→ Preparation :- The JVM allocates memory for the static fields of a class or interface and initializes them with default values

→ Resolution :- Symbolic references are replaced with direct References in the runtime Constant pool

* Initialization

Initialization involves executing the initialization method of the class or interface this can include calling the class Constructor executing the static block and assigning values to all the static variables

② Runtime Data Area

* Method Area

load all the class information used to keep type information about the classes the method area is created on the Virtual machine start-up and there is only one method area per JVM

* Heap Area

This is the run time data area from which memory for all class instances and array is allocated there is only one heap area per JVM

* Stack Area

whenever a new thread is created in the JVM a seperate runtime Stack is also created at the same time All local variable method calls and

```
}
}
```

O/P Arithmetic Operation
a=2
b=6
c=1
d=-1
e=i

It also include

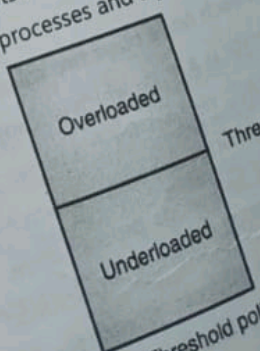
%	Modulus // a % b
++	Increment // a++
--	decrement // a--
+=	Addition assignment
-=	Subtraction assignment
/=	division assignment
%=	Modulus assignment
*=	Multiplication assignment

Bitwise Operator
These operator act upon the individual bit of there operand they are

~	Bitwise unary NOT
&	Bitwise AND
|	Bitwise OR
^	Bitwise Exclusive OR
>>	Shift right
>>>	Shift right zero fill
<<	Shift left

policy.

Most of the load-balanc
Overloaded and underloaded. If
underloaded and it accepts new local or remo
nodes transfer the local processes and reject to accept



Single-Threshold policy

(a)

tely.

icy.

n particular node.

n migrate from one node to other.

If process is processed on node which is not
node and admitted for processing then
ss the network. It then is

output

a = 0011

b = 0110

a|b = 0111

a&b = 0010

a^b = 0101

~a&b | a&~b = 0101

~a = 1100

Relational operator

Determine the relationship that one operand has to the other

operand Any type

Return :- Boolean value

==	Equal to
!=	Not Equal to
>	greater than
<	less than
>=	Greater than or equal to
<=	less than or equal to

Ex Class Relate {

public static void main (String args[]).

{

int a=4;

int b=1;

boolean c=a<b;

System.out.println (c);

}

}

Q5 what are the primitive data type in java? Briefly Explain their size range and other details

Ans primitive data types (8 types)

① Integer

This group includes byte, Short, int and long which are for whole valued signed numbers All of these are signed, positive and negative values.

long

Size = 64 bits

range :- -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

use :- useful for those occasion where an int type is not large enough to hold the desired value

Int

Size :- 32 bits

range = -2,147,483,648 to 2,147,483,647

uses : when byte and Short values are used in an expression, they are promoted to int when the expression is evaluated So int is the most commonly used integer type It is the best choice when an integer is needed

Store as double precision

② characters

char

Size : 16 bits

Range : 0 to 65,536 (no negative chars)

uses : Represent a single character or it stores the characters char in java is designed to hold unicode character, it can also be used as an integer type

④ Boolean

Range : True or false

Uses : It can have only one of two possible values, true or false this is the type returned by all relational operator, as in the case of a<b boolean is also the type required by the conditional expression that govern the control statments such as if and for

Q6 Explain about memory management in java with reference to stack and heap

Ans The process of allocation and deallocation of object is called memory management the JVM divides the memory into two parts stack memory and heap memory

Stack

Stack memory is a physical space allocated to each thread at run time it is created when a thread creates memory management in the stack follows LIFO order because it is accessible globally It stores the variable reference to object and partial result

feature

⇒ Variable inside the stack exist only as long as the method that created them is running

⇒ It is automatically allocated and deallocated when the method finishes execution

⇒ Access to this memory is fast when compared to heap memory

⇒ If this memory is full java throws java.lang StackOverflowError

⇒ This memory is threadsafe as each thread operates in its own stack

Heap

It is created when the JVM start-up and used by the application as long as the application runs It stores Object and JRE classes whenever we create object it occupies space in the heap memory while the reference of that object create is the stack It dynamically handles the memory the memory manually java provides the garbage collector that deletes the object which are no longer being used

features

→ If heap space is full, java throws java.lang.OutOfMemoryError

→ Access to this memory is comparitively slower than stack memory

→ This memory in contrast to stack isn't automatically deallocated It needs garbage collector to free up unused object so as to keep the efficiency of the memory usage

byte → short → char → int → long → float → double

Ex class Main {
    public static void main (String arg [])
    {
        int i = 9;
        double d = i;
        System.out.println (i);
        System.out.println (d);
    }
}

output 9
9.0

→ It is also known as implicit conversion
→ It is done automatically
→ It is safe because there is no chance to lose data It takes place when
* Both data type must be compatible with each other
* the target type must be longer than source type

→ It is specified using the keyword 'private'
→ The methods or data members declared as private are accessible only within the class in which they are declared
→ Any other class of the same package will not be able to access these member
→ top-level classes or interfaces can not be declared as private

④ Protected
→ It is specified using the keyword 'protected'
→ The method or data members declared as protected are accessible within the same package or subclasses in different packages
→ It provides more accessibility than default modifier

Q1 Write a short note on access specifier in Java

Ans Access specifier help to restrict the scope of the class constructor, variable method or data member It provides security, accessibility etc be the user depending upon the access specifier used with the element

① Default

when no access specified is used for a class, method or data member It is said to be having the default access specifier by default the data member classes, or methods that are not declared using any access modifier i.e having default access specifier are accessible only within the same package

② Public

The public access specifier is used using the keyword 'public'

→ The public access specifier has all the widest scope among all other specifiers

→ classes, methods or data member that are declared as public are accessible from anywhere in the program there is no restriction on the scope of the public data members

③ public

→ A Static method can be invoked without the need for Creating an instance of a class
→ A Static method belongs to the class rather than the object of a class
→ A Static method can access Static data member and can change the value of it
→ The most common example of a Static method is the main () method
→ They cannot refer to this or Super in any way

Classes

→ A class can be made static only if it is a nested class
→ Nested Static class doesn't need a reference of outer class In this case a Static class cannot access non-Static member of the outer-class

Block

→ It is used to initialize the Static data member
→ It is executed before the main method of the time of class loading

Q6 Write in detail about Static Keyword

Ans The Static Keyword in Java is mainly used for memory management. The Static Keyword in Java is used to share the same variable or method of a given class the Static Keyword belongs to the class than an instance of the class the Static Keyword is used for a constant variable or a method that is the same for every instance of a class It refers to the common property of all objects The Static Keyword can be applied

Variable

→ when a variable is declared as Static it is known as Static variable
→ The Static variable gets memory only once in the class area at the time of class loading
→ It makes program memory efficient
→ Static variables are essentially global variables are essentially global variables
→ Static variables are created at class level only

Method

→ when Static Keyword is applied with any method it is known as Static method

Q1 Explain the term narrowing widening

Ans Narrowing
Converting a larger type to smaller size type
double → float → long → int → char → short → byte
Narrowing must be done manually by placing the type in parenthesis in front of the value

Ex. class Main {
public static void main (String args [])
{
double d = 9.78d;
int i = (int) d
System.out.println (d);
System.out.println (i);
}
}
output : 9.78
9
→ Also known as explicit conversion
→ Converting higher data type to lower data type

Widening
Converting a smaller type to a larger type size

short
Size : 16 bits
range : -32,768 to 32,767
uses : least and java type longer than byte

Byte
Size : 8 bits
range : -128 to 127
uses : used while working with stream of file or network and with raw binary data

② Floating point types
Also known as real numbers are used when evaluation expression that require fractional precision
double
Size : 64 bits
range : 4.98 · 324 to 1.80 308
uses : It is usually faster than single precision on some modern processor that have been optimized for high speed mathematical calculation
float
Size : 32 bits
range : 1.40.045 to 3.48 +038
uses : It specifies the single precision value that uses 32 bits of storage. Single precision is faster on some processor and takes half as much

```
System.out.println("b = "+b);
System.out.println("a|b = "+c);
System.out.println("a&b = "+d);
System.out.println("a^b = "+e);
System.out.println("!a&b|a&!b = "+f);
System.out.println("!a = "+g);
}
}
```

a = true
b = false
a|b = true
a&b = false
a^b = true
!a&b|a&!b = true
!a = false

Ternary operator
Replace certain type of if then else statment
operand : Boolean expression
Return : Based on operand (any)
expression? expression 2 : expression 3

Ex

```
int x = 10, y = 15
int result = (x>y) ? x : y
System.out.println(result);
```

output :- 15

PAGE NO.
DATE / /

* output :- false
logical operators
perform logical operation on operand
operand : only Boolean type
Return Boolean :

OP	Result
&	logical AND
|	logical OR
^	logical XOR
||	Short-Circuit OR
&&	Short-Circuit AND
!	logical unary NOT
&=	AND Assigment
|=	OR assigment
^=	XOR assignment
==	Equal to
!=	Not Equal to
?	Ternary if-then-else

```
class BoolOpr {
public static void main (String args[]); {
    boolean a = true;
    boolean b = false;
    boolean c = a|b;
    boolean d = a&b;
    boolean e = a^b;
    boolean f = (!a&b) | (a&!b);
    boolean g = !a
System.out.println ("a = "+a);
```

&=	Bitwise AND assignment
|=	Bitwise OR Assignment
^=	Bitwise exclusive OR assignment
>>=	Shift right assignment
>>>=	Shift right zero fill assignment
<<=	Shift left assignment

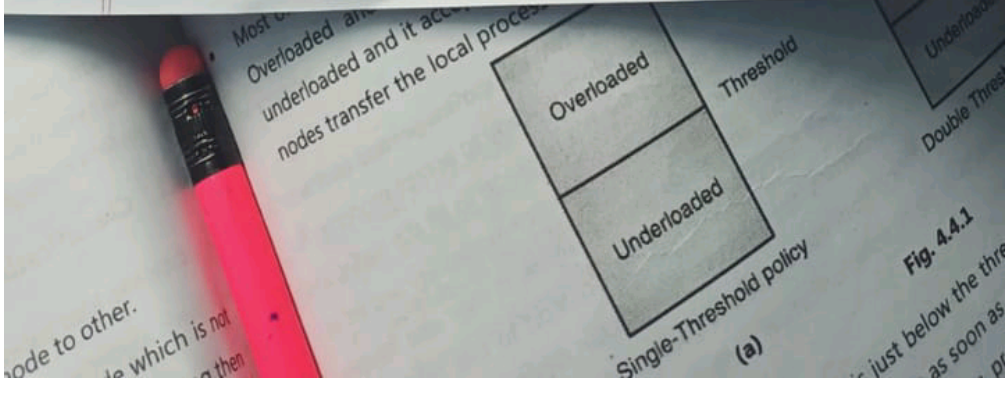
Operands :- Integer type

```
class Bitlogic {
    public static void main (String args[]) {
    String binary[] = { "0000", "0001", "0010",
"0011", "0100", "0101", "0110", "0111", "1000", "1001",
"1010", "1011", "1100", "1101", "1110", "1111" };
int a = 3
int b = 6
int c = a|b
int d = a&b
int e = a^b
int f = ((~a)&b) | (a&(~b))
int g = ~a & a xor;
System.out.println("a = " + binary[a]);
System.out.println("b = " + binary[b]);
System.out.println("a|b = " + binary[c]);
System.out.println("a&b = " + binary[d]);
System.out.println("a^b = " + binary[e]);
System.out.println("~a&b|a&~b = " + binary[f]);
System.out.println("~a = " + binary[g]);
}
}
```

Q4 write in detail about different types of operator in java, Category wise quantity their functionality operand and return type Give one Example Statment for each

Ans Arithmetic Operator
The basic Arithmetic Operator - addition, Subtraction, multiplication and division all behave as expected for all numeric type the unary minus operator negates its single operand the unary plus operator simply returns the value of its operand
operand :- int, float, double etc (numeric)
Return type : arithmetic

```
Class BasicMath {
    public static void main (String args [ ]) {
    System.out.println ("Arithmetic Operator");
        int a = 1+1;
        int b = a*3;
        int c = b/4;
        int d = c-a;
        int e = -d;
        System.out.println ("a = "+a);
        System.out.println ("b = "+b);
        System.out.println ("c = "+c);
        System.out.println ("d = "+d);
        System.out.println ("e = "+e);
```



PAGE NO.
DATE / /

→ Garbage Collector
Collects and removes unreferenced object from the heap area. It is the process of reclaiming the runtime unused memory automatically by destroying them.
* Mark : GC identifies the Unused Object
* Sweep : removes the objects identified.

④ Java Native Interface
It is necessary to use native code this can be in cases where we need to interact with hardware or to overcome the memory management and performance constraints in Java. Java supports the Execution of native code via the Java Native Interface

⑤ Native Method libraries
Native Method libraries are libraries that are written in other programming languages such as C, C++, and assembly these libraries are usually present in the form of ".dll", or ".so" files these native libraries can be loaded through JNI.

estimate the workload of the particular node based on some measur
which are time dependent and node-dependent. These factors ar
of load estimation, resource demands of these processes
processors.
in efficient manner. Total
load. This is

In single thresho
case, load beco
the remote p
unstable.

partial results are stored in the Stack area

* Program Counter Register
The JVM supports multiple threads at the same time. Each thread has its own PC register to hold the address of the currently executing JVM instructions once the instruction is executed the PC register is updated with the next register

* Native Method Stacks
The JVM contains stack that support native method these method are written in a language other than Java such as C and C++ for every new thread a separate native method stack is also allocated

② Execution Engine
Once the byte code has been loaded into the main memory and details are available in the runtime data area, the next step is to run the program the Execution Engine handles this by executing the code present in each class
→ Interpreter :- The Interpreter reads and execute the bytecode instruction line by line.
→ JIT Compiler :- The JIT compiler overcomes the disadvantage of the interpreter when the interpreter finds same repeated code it uses the JIT compiler